IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Patent Application

Inventors: Peter Joseph Giacomini et al.

Serial No.: 09/725737

Conf. No.: 9624

Filing Date: 11/29/2000

Art Unit: 2142

Examiner: Thong H Vu

Docket No.: 500-002US

Title: Method and Apparatus For Economical Cache Population

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Enclosed are the following papers related to the above-identified patent application:

- Transmittal Letter
- Appeal Brief

Pursuant to 37 CFR 1.136(a)(3), please treat this and any concurrent or future reply in this application that requires a petition for an extension of time for its timely submission as incorporating a petition for extension of time for the appropriate length of time.

Respectfully,
Peter Joseph Giacomini et al.

By **/Jason Paul DeMont/**
Jason Paul DeMont
Reg. No. 35793
Attorney for Applicants
732-578-0103 x11

DeMont & Breyer, L.L.C.
Suite 250
100 Commons Way
Holmdel, NJ 07733
United States of America

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Patent Application

Inventors: Peter Joseph Giacomini et al.

Serial No.: 09/725737

Conf. No.: 9624

Filing Date: 11/29/2000

Art Unit: 2142

Examiner: Thong H Vu

Docket No.: 500-002US

Title: Method and Apparatus For Economical Cache Population

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

APPEAL BRIEF UNDER 37 CFR 41.67

Pursuant to 37 CFR 41.67, this brief is filed in support of the appeal in this application.

TABLE OF CONTENTS

REAL PARTY IN INTEREST

The real party of interest in this application is the assignee of this application: Broadspider Networks, Inc. The applicant's attorneys, Jason Paul DeMont and Wayne S. Breyer, are shareholders of Broadspider Networks, Inc.

RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences.

STATUS OF CLAIMS

Claims 1 through 32 stand rejected and are being appealed.

STATUS OF AMENDMENTS

All amendments have been entered. There have been no amendments made subsequent to the final rejection.

SUMMARY OF THE CLAIMED SUBJECT MATTER

The present invention relates to data processing systems and computer networks in general, and, more particularly, to techniques for storing resources in a cache.

In the context of the Internet, when a user of the World Wide Web requests a Web page, the user must wait until the page is available on his or her data processing system (*e.g.*, computer, *etc.*) for viewing. In general, this wait occurs because the request for the Web page must traverse the Internet from the user's data processing system to the data processing system that is the source of the page, the request must be fulfilled, and the requested page must travel back to the user's system. If the Internet is congested or the data processing system that is the source of the page is overwhelmed, the wait can be considerably long. **[Specification Page 1, Lines 10-16]**

To shorten this wait, special data processing systems are deployed throughout the Internet to expedite the delivery of some Web pages. Some of these data processing systems expedite the delivery of Web pages by functioning as cache memories, which are also known as "caches." For example, a cache stores commonly requested Web pages and thereafter intercepts and fulfills requests for those pages, which eliminates the need for the request having to travel to, and be serviced by, the principal memory. **[Specification Page 1, Lines 17-22]**

A cache expedites the delivery of the Web page in two ways. First, a cache eliminates the need for the request to travel all of the way to the system that is the ultimate source of the page, and, therefore, eliminates some of the wait associated with the transit. Second, a cache also reduces the number of Web page requests that must be fulfilled by the system that is the ultimate source of the page, and, therefore, the wait associated with contention for the system is eliminated. **[Specification Page 1, Lines 18-27]**

Caches are also used to expedite processing in data processing systems and their operation in data processing systems is roughly analogous to, with notable exceptions, their operation in computer networks. At this time, the applicants do not believe that the differences between the use of cache in computer networks and in data processing systems are germane to the present appeal and the scope of the pending claims encompass the use of the present invention in both computer networks and data processing systems.

When a cache is full and an additional resource is to be stored in the cache, the cache must have an algorithm for deciding which resource gets purged to make room for the additional resource. There is a lot of prior art in this area.

In contrast, there is not a lot of prior art addressing the issue of when to store the resource in the cache in the first place. Typically, a resource is stored in a cache either (1) the first time that it is requested, or (2) it is pre-fetched before the first time that it is requested. The present invention addresses the issue of when to store a resource in a cache that is particularly efficient in many cases.

For example, a cache in accordance with the present invention is populated with a resource only when at least *i* requests for the resource have been received, wherein *i* is an integer and is, at least occasionally, greater than one. For example, a cache in accordance with the illustrative embodiment might not be populated with a resource unless two requests for the resource have been received within 10 minutes. In general, embodiments of the present invention are advantageous because *they prevent the cache from being populated with infrequently requested resources*. **[Specification Page 3, Lines 21-31]**

Here's a way to keep all of these ideas straight in one mind. There are only three ways that a cache can populated:

- **Pre-Filling** (AKA Pre-Fetching & Pre-Loading) — Populating a cache with a resource **before** a request for the resource has arrived.
- **Post-Filling** (AKA Responsive Loading & Responsive Caching) — Populating a cache with a resource after one request for the resource has arrived.
- **Delayed Post-Filling** — Populating a cache with a resource after more than one request for the resource has arrived.

Pre-filling and post-filling are old and well-known in the art. The present invention can be thought of as delayed post-filling and the combination of post-filling and delayed post-filling.

The present application comprises four (4) independent claims. Each shall be presented, summarized, and mapped to the specification by page and line number and to the drawings, if any.

Independent claim 1 recites:

> **1.** (Previously Presented) A method comprising:
>
> populating a cache with a resource only when at least *i* requests for said resource have been received;
>
> wherein *i* is an integer and is at least occasionally greater than one.

Claim recites a method of delayed post-filling, and is described in the specification at **Page 9, lines 19-20; Page 13, lines 1-4; and drawings at Figure 6, boxes 608 and 609**.

Independent claim 8 recites:

> **8.** (Previously Presented) A data processing system comprising:
>
> a cache for storing a resource; and
>
> a processor for populating said cache with said resource only when at least *i* requests for said resource have been received;
>
> wherein *i* is an integer and is at least occasionally greater than one.

Claim 8 recites a data processing system that performs delayed post-filling, and is described in the specification at **Page 7, line 11; Page 9, lines 19-20; Page 13, lines 14-18 and drawings at Figure 5, boxes 502 and 503**.

Independent claim 15 recites:

> **15.** (Previously Presented) A method comprising:
>
> receiving at a first node in a computer network at least one request for a resource;
>
> retrieving said resource from a second node in said computer network; and
>
> populating a cache in said first node with said resource only when at least *i* requests for said resource have been received at said first node;
>
> wherein *i* is an integer and is at least occasionally greater than one.

Claim 15 recites a method of delayed post-filling, and is described in the specification at **Page 4, lines 21-23; Page 9, lines 19-20; Page 14, lines 1-6; and drawings at Figure 4, boxes 411 and 423; and Figure 6, boxes 608 and 609**.

Independent claim 24 recites:

> **24.** (Previously Presented) A first node in a computer network, said first node comprising:
>
> a cache;
>
> at least one receiver for receiving at least one request for a resource; and
>
> a processor for retrieving said resource from a second node in said computer network, and for populating said cache in said first node with said resource only when at least *i* requests for said resource have been received at said first node;
>
> wherein *i* is an integer and is at least occasionally greater than one.

Claim 24 recites a node in a computer network that comprises hardware for populating a cache in accordance with delayed post-filling, and is described in the specification at **Page 7, line 11; Page 9, lines 19-20; Page 15, lines 1-7, and drawings at Figure 5, boxes 502 and 503.**

GROUNDS OF OBJECTION AND REJECTION TO BE REVIEWED ON APPEAL

Ground 1: 35 U.S.C. 112 Rejection of Claims 2, 9, 16, and 25

Claims 2, 9, 16, and 25 were rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement.

Ground 2: 35 U.S.C. 102 Rejection of Claims 1, 8, 11, 12, 15, 22-24, 27-28, and 31-32

Claims 1, 8, 11, 12, 15, 22-24, 27-28, and 31-32 have been rejected under 35 U.S.C. 102(e) as being anticipated by V. Tran et al., U.S. Patent 7,039,683 B1 (hereinafter "**Tran**").

Ground 3: 35 U.S.C. 102 Rejection of Claims 1, 8, 11, 12, 15, 22-24, 27-28, and 31-32

Claims 1, 8, 11, 12, 15, 22-24, 27-28, and 31-32 have been rejected under 35 U.S.C. 102(e) as being anticipated by J.T. Chamberlain et al., U.S. Patent 6,408,360 B1 (hereinafter "**Chamberlain**").

Ground 3: 35 U.S.C. 102 Rejection of Claims 1, 8, 11, 12, 15, 22-24, 27-28, and 31-32

Claims 1, 8, 11, 12, 15, 22-24, 27-28, and 31-32 have been rejected under 35 U.S.C. 102(e) as being anticipated by D. Teoman et al., U.S. Patent 6,564,509 B1 (hereinafter "**Teoman**").

ARGUMENTS

Ground 1: 35 U.S.C. 112 Rejection of Claims 2, 9, 16, and 25

Claims 1, 8, 15, and 24 were rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. The applicants respectfully traverse the rejection.

The Office action states:

> i.e., *i* is an integer and greater than one (=two or more, or variable, claims 1, 8 , 15, 24) versus *i* is invariant (claims 2, 9 , 16, 25). The specification teaches these as alternative embodiments since claim 1 clearly recited is variable by reciting the *i* is occasionally [sic] greater than 1. It can't be invariant.
>
> Office action of 4/27/07, page 4, first paragraph

The applicants respectfully submit that the Office is confused.

Claim 1 recites:

> **1.** A method comprising:
>
> populating a cache with a resource only when at least *i* requests for said resource have been received;
>
> wherein *i* is an integer and ***is at least occasionally greater than one***.
>
> (*emphasis supplied*)

The Office is incorrectly interpreting the language "is ***at least*** occasionally greater than one" to mean "that *i* ***varies*** and there ***must*** be an occasion when *i* is ***not*** greater than one." Clearly this is incorrect.

The language "is ***at least*** occasionally greater than one" means "there is at least one occasion when *i* is greater than one and there ***might*** be, but there is ***not*** necessarily, one or more occasions when ***i*** is **not** greater than one." For example, when *i* = 2 and is fixed, *i* is at least occasionally greater than one.

Claim 2 recites:

> **2.** The method of claim 1 wherein the value of *i* is ***invariant***.
>
> (*emphasis supplied*)

Because the Office interpreted the language in claim 1 to mean that *i* varies, and the language in claim 2 clearly states that it does not vary, the Office has rejected claim 2 as incongruous with claim 1. But because the Office's interpretation of claim 1 is incorrect,

and because claim 1 does not require that *i* vary, the applicants respectfully submit that the rejection of claim 2 is traversed.

The logic underlying the Office's rejection of claims 9, 16, and 25 is identical, and the applicants respect that the rejected of them is also traversed for the same reasons.

Ground 2: 35 U.S.C. 102 Rejection of Claims 1, 8, 11, 12, 15, 22-24, 27-28, and 31-32

Claims 1, 8, 11, 12, 15, 22-24, 27-28, and 31-32 have been rejected under 35 U.S.C. 102(e) as being anticipated by V. Tran et al., U.S. Patent 7,039,683 B1 (hereinafter "**Tran**"). The applicants respectfully traverse.

The 102(e) filing date of Tran is 29 December 2000, which is **after** the filing date of the present application (29 November 2000), and, therefore, it is a invalid prior art reference.

For this reason, the applicants respectfully submit that the rejection is traversed.

Ground 3: 35 U.S.C. 102 Rejection of Claims 1, 8, 11, 12, 15, 22-24, 27-28, and 31-32

Claims 1, 8, 11, 12, 15, 22-24, 27-28, and 31-32 have been rejected under 35 U.S.C. 102(e) as being anticipated by J.T. Chamberlain et al., U.S. Patent 6,408,360 B1 (hereinafter "Chamberlain"). The applicants respectfully traverse.

Claim 1 recites:

> **1.** (Previously Presented) A method comprising:
>
> populating a cache with a resource only when at least *i* requests for said resource have been received;
>
> ***wherein i is an integer and is at least occasionally greater than one***.
>
> (*emphasis supplied*)

Nowhere does Chamberlain teach or suggest, alone or in combination with the other references, what claim 1 recites — namely, delaying the population of a cache with a resource until at least *i* requests for the resource have been received, ***wherein i is at least occasionally greater than one***. In other words, claim 1 recites a form of delayed post filling.

To support the rejection, the Office cites Column 14, lines 19-26, which states:

> The applicability analysis portion of the cached-response analysis only examined, as an example, one strategy flag (OnlyAnonymous). However, there are other request-specific characteristics that could as easily be tested. Tests for appropriate browser type and version, and tests for the appropriate language are examples of other user-specific tests that may be run against a cached response to ensure that it is applicable to the request or the requesting user.
>
> Chamberlain Col. 14, lines 19-26

The applicants respectfully submit that there is nothing — absolutely nothing — that is relevant to the issue at hand — namely the conditions under which a cache is populated with a resource. For this reason, the applicants respectfully submit that the rejection of claim 1 is traversed.

Because claim 8 depends on claim 1, the applicants respectfully submit that the rejection of it is also traversed.

Independent claims 8, 15, and 24 all also recite the limitation "***wherein i is an integer and is at least occasionally greater than one***," and the Office reads the same portion of Chamberlain onto this limitation. For this reason, the applicants respectfully submit that the rejection of them is also traversed.

Because claims 12, 22-23, 27-28, and 31-32 depend on one of these independent claims, the applicants respectfully submit that the rejection of them is also traversed.

Ground 4: 35 U.S.C. 102 Rejection of Claims 1, 8, 11, 12, 15, 22-24, 27-28, and 31-32

Claims 1, 8, 11, 12, 15, 22-24, 27-28, and 31-32 have been rejected under 35 U.S.C. 102(e) as being anticipated by D. Teoman et al., U.S. Patent 6,463,509 B1. The applicants respectfully traverse.

Claim 1 recites:

> **1.** (Previously Presented) A method comprising:
>
> populating a cache with a resource only when at least *i* requests for said resource have been received;
>
> ***wherein i is an integer and is at least occasionally greater than one***.
>
> (*emphasis supplied*)

Nowhere does Teoman teach or suggest, alone or in combination with the other references, what claim 1 recites — namely, delaying the population of a cache with a resource until at least *i* requests for the resource have been received, ***wherein i is at least occasionally greater than one***.

To support the rejection, the Office cites Column 9, lines 15-38, which states:

> In various embodiments discussed below, the user cache manager 90 generates a user interface on a display of the host computer system to allow a user to configure the operation of the user cache 25. In general, ***there are two types of storage operations that take place in the user cache***: ***preloading*** and ***responsive caching***. ***Responsive caching refers to the storage of data in the user cache in response to I/O requests from application processes. In a preload operation, by contrast, data is retrieved from mass storage and stored in the user cache before being requested for use in an application process***. In one type of preloading, called commanded preloading, data is preloaded according to user-provided criteria. For example, the user may command, via a user interface presented by the user cache manager 90, that specific files, specific file types or files from a specific source such as a disk drive or subdirectory be loaded into the cache, even before the files are requested by application processes. This is shown in FIG. 5 by the user preferences supplied to the user cache manager. In another type of preloading, called triggered

> preloading, data is preloaded when the access statistics 94 gathered by the observer 95 indicate that a user-specified threshold is exceeded. As discussed below, numerous user supplied parameters may be used to control commanded and dynamic preloading operations.
>
> Teoman Col. 9, lines 15-38 (*emphasis supplied*)

A careful reading of this portion of Teoman teaches exactly what the applicants have contended is the prior art before the present invention — that there were two types of cache population: pre-loading (AKA pre-filling and pre-fetching) and responsive caching (AKA post-filling). Teoman goes on to explain both types of cache population, but nowhere teaches or suggests delayed post-filling as recited in the claim.

A second portion of Teoman relied on by the Office is at Column 15, lines 24-50, which states:

> ***Examples of preloading policies include***, but are not limited to, preloading complete files in response to file segment access, preloading all files within the directory or folder of a launched application, ***preloading all files in a directory or folder if a threshold number of files from the directory or folder have already been accessed***, preloading files in the system directory or folder, preloading files having a particular file type identifier if a threshold number of files having the file type identifier have been accessed, and so forth. The file type identifier may be a filename extension such as ".doc" or ".psd", as used in many operating systems, or the file type identifier may be a file attribute that does not appear in the file name. Also, as indicated in FIG. 10, the threshold number of files that have a particular file type identifier and the threshold number of files that are from a directory or folder are specified by the user. In many cases, the preloading policies translate directly into criteria for triggered preloading. For example, a policy to preload all files in a directory or folder if a threshold number of the files have been accessed sets up a preload trigger. The user cache manager periodically inspects the access table maintained by the observer to determine if the trigger criteria is met (e.g., whether the threshold number of files from the indicated directory have been accessed). Other preload policies give rise to commanded preload operations. For example, a policy to preload the system

> directory causes the user cache manager to begin commanded preloads of the directory contents.
> Teoman Col. 15, lines 24-50(*emphasis supplied*)

Here, Teoman is teaching a variety of tests for pre-loading a resource into a cache ***before the resource is requested***. Designing a good test for pre-loading is difficult, and Teoman teaches several good tests. But these are all tests for ***pre-loading*** a resource into a cache ***before the resource is requested***. Nowhere does Teoman teach or suggest delayed-post filing. For this reason, the applicants respectfully submit that the rejection of claim 1 is traversed.

Because claim 8 depends on claim 1, the applicants respectfully submit that the rejection of it is also traversed.

Independent claims 8, 15, and 24 all also recite the limitation "***wherein i is an integer and is at least occasionally greater than one***," and the Office reads the same portion of Teoman onto this limitation. For this reason, the applicants respectfully submit that the rejection of them is also traversed.

Because claims 12, 22-23, 27-28, and 31-32 depend on one of these independent claims, the applicants respectfully submit that the rejection of them is also traversed.

CONCLUSION

The applicants have demonstrated that the logic underlying the Office's rejection is untenable, and, therefore, that the rejection is not sustainable. For this reason, the applicants respectfully request the Board of Appeals to reverse the decision of the Examiner as provided for in 37 C.F.R. 41.50(a).

Respectfully,
Peter Joseph Giacomini et al.

By **/Jason Paul DeMont/**
Jason Paul DeMont
Reg. No. 35793
Attorney for Applicants
732-578-0103 x11

DeMont & Breyer, L.L.C.
Suite 250
100 Commons Way
Holmdel, NJ 07733
United States of America

CLAIMS APPENDIX

1. (Previously Presented) A method comprising:

populating a cache with a resource only when at least *i* requests for said resource have been received;

wherein *i* is an integer and is at least occasionally greater than one.

2. (Original) The method of claim 1 wherein the value of *i* is invariant.

3. (Original) The method of claim 1 wherein the value of *i* is based on calendrical time.

4. (Original) The method of claim 1 wherein said cache is populated with said resource only when at least *i* requests for said resource have been received within an elapsed time interval, Δt.

5. (Original) The method of claim 4 wherein the duration of said elapsed time interval, Δt, is based on the value of *i*.

6. (Original) The method of claim 4 wherein the value of *i* is based on calendrical time.

7. (Original) The method of claim 4 wherein the duration of said elapsed time interval, Δt, is based on calendrical time.

---

8. (Previously Presented) A data processing system comprising:

a cache for storing a resource; and

a processor for populating said cache with said resource only when at least *i* requests for said resource have been received;

wherein *i* is an integer and is at least occasionally greater than one.

9. (Original) The data processing system of claim 8 wherein the value of *i* is invariant.

10. (Original) The data processing system of claim 8 wherein the value of *i* is based on calendrical time.

11. (Original) The data processing system of claim 8 wherein said cache is populated with said resource only when at least *i* requests for said resource have been received within an elapsed time interval, Δt.

12. (Original) The data processing system of claim 8 wherein the duration of said elapsed time interval, Δt, is based on the value of *i*.

13. (Original) The data processing system of claim 8 wherein the value of *i* is based on calendrical time.

14. (Original) The data processing system of claim 8 wherein the duration of said elapsed time interval, Δt, is based on calendrical time.

---

15. (Previously Presented) A method comprising:

receiving at a first node in a computer network at least one request for a resource;

retrieving said resource from a second node in said computer network; and

populating a cache in said first node with said resource only when at least *i* requests for said resource have been received at said first node;

wherein *i* is an integer and is at least occasionally greater than one.

16. (Original) The method of claim 15 wherein the value of *i* is invariant.

17. (Original) The method of claim 15 wherein the value of *i* is based on calendrical time.

18. (Original) The method of claim 15 wherein said cache is populated with said resource only when at least *i* requests for said resource have been received within an elapsed time interval, Δt.

19. (Original) The method of claim 18 wherein the duration of said elapsed time interval, Δt, is based on the value of *i*.

20. (Original) The method of claim 18 wherein the value of *i* is based on calendrical time.

21. (Original) The method of claim 18 wherein the duration of said elapsed time interval, Δt, is based on calendrical time.

22. (Original) The method of claim 15:

wherein said computer network is a hierarchical computer network and said first node has *m* filial nodes;

wherein said cache is populated with said resource only when at least one request for said resource has been received from at least *n* of said *m* filial nodes; and

wherein *m* is an integer greater than one, *n* is an integer greater than one, and $m \geq n$.

23. (Original) The method of claim 15:

wherein said computer network is a hierarchical computer network and said first node has *m* filial nodes;

wherein said cache is populated with said resource only when at least one request for said resource has been received from at least *n* of said *m* filial nodes within an elapsed time interval, Δt; and

wherein *m* is an integer greater than one, *n* is an integer greater than one, and $m \geq n$.

---

24. (Previously Presented) A first node in a computer network, said first node comprising:

a cache;

at least one receiver for receiving at least one request for a resource; and

a processor for retrieving said resource from a second node in said computer network, and for populating said cache in said first node with said resource only when at least *i* requests for said resource have been received at said first node;

wherein *i* is an integer and is at least occasionally greater than one.

25. (Original) The first node of claim 24 wherein the value of *i* is invariant.

26. (Original) The first node of claim 24 wherein the value of *i* is based on calendrical time.

27. (Original) The first node of claim 24 wherein said cache is populated with said resource only when at least *i* requests for said resource have been received within an elapsed time interval, Δt.

28. (Original) The first node of claim 27 wherein the duration of said elapsed time interval, Δt, is based on the value of *i*.

29. (Original) The first node of claim 27 wherein the value of *i* is based on calendrical time.

30. (Original) The first node of claim 27 wherein the duration of said elapsed time interval, Δt, is based on calendrical time.

31. (Original) The first node of claim 24:

wherein said computer network is a hierarchical computer network and said first node has *m* filial nodes;

wherein said cache is populated with said resource only when at least one request for said resource has been received from at least *n* of said *m* filial nodes; and

wherein *m* is an integer greater than one, *n* is an integer greater than one, and $m \geq n$.

32. (Original) The first node of claim 24:

wherein said computer network is a hierarchical computer network and said first node has *m* filial nodes;

wherein said cache is populated with said resource only when at least one request for said resource has been received from at least *n* of said *m* filial nodes within an elapsed time interval, Δt; and

wherein *m* is an integer greater than one, *n* is an integer greater than one, and $m \geq n$.

EVIDENCE APPENDIX

There is no evidence submitted pursuant to 37 CFR §§ 1.130, 1.131, or1.132.

RELATED PROCEEDINGS APPENDIX

There are no related proceedings.